Visualization Verification and Benchmark Suites

Al Globus, Computer Sciences Corporation

Report RNR-91-028
7 October 1991

Position paper for the Visualization Environments Workshop at Visualization '91, October 1991

This work was supported by NASA Contract NAS 2-12961 to Computer Sciences Corporation for the Numerical Aerodynamic Simulation Systems Division at NASA Ames Research Center.

Copies of this report are available from:

NAS Applied Research Office
Mail Stop T045-1
NASA Ames Research Center
Moffett Field, CA 94035
(415) 604-4332

Visualization '91 Scientific Visualization Environments Position Paper
Visualization Verification and Benchmark Suites

Al Globus, Computer Sciences Corporation

There are a number of visualization environments today: FAST [BANC90], AVS [UPSON89], SGI's Explorer, ApE [DYER90], Flora [HULT91], Wavefront's Data Visualizer, etc. Most of these environments generate particle traces, isosurface, scalar mapped cutting planes and a few other more or less standard visualizations. One might like to compare these systems.

One can compile lists of features and/or compare visualization environments analytically using the vector-bundle concepts David Butler [BUTL89] has developed. One can use data sets developed for a scientific discipline and compare results and performance. These data sets are designed to investigate phenomena, not evaluate software. Thus, correctness is difficult to verify and different aspects of performance are difficult to isolate. I argue that, in addition to these approaches, a visualization benchmark/verification suite could be of great value.

Correctness is vital for acceptance by critical users and will become even more important as visualization software provides quantitative data (e.g., isosurface area, streamline length, etc.). I have used a number of visualization environments. Other than blind faith, I have no reason to believe the results are more than approximately accurate most of the time. Worse, there is no easy way to quantify error, which is always present to some degree. I have noticed subjective differences in the results using different packages, but I have no straightforward means of determining how much the results vary, which is more nearly correct, or if either is even close. This is unacceptable. Furthermore, developers testing systems have no well established means of determining the correctness of their systems. They are left with the ad hoc approaches of the development teams.

One may compare the performance of different packages on data sets developed by various application fields, but the results may be difficult to interpret if your data do not have nearly identical characteristics. A well designed set of benchmarks could conceivably allow one to characterize various aspects of visualization package performance; e.g., it has very fast streamlines but slow isosurface generation, interpolation speed is mediocre but isosurface representation is very compact. High performance computing has many years of experience designing and examining the results of benchmarks. We could do worse than follow their lead.

The NAS facility where I work is concerned with computational fluid dynamics (CFD). Correctness is a particularly important issue in CFD since computational grids may contain complex topologies including multiple blocks and iblanking. Iblanking refers to a solver technique in which some points in a curvilinear (warped) grid are masked out; i.e., are not considered valid. This complicates visualization techniques since one must avoid visualizing the masked data points. Furthermore, grids may contain singularities where adjacent grid points have the same physical space coordinates. Visualization code may not be considered correct unless it can properly handle the many special cases that arise due to masking, multiple block grids, and grid singularities.

CFD data sets are legendary for their size. The current state of the art in CFD solutions is multiple three dimensional grids with a few million points and (for unsteady flows) a few thousand time steps. Each grid point is represented by three floating point numbers (location), and five floating point numbers per time step for the solution. No visualization environment on any hardware can handle these data sets today, although some function adequately on a single time step. Nor, as a general rule, can the developers tell you what the size limitations to expect given the hardware limitations. A set of variable sized benchmarks could provide a means for developers to discover

and communicate the size limitations of their products.

A good set of benchmarks should have at least the following properties (in no particular order):

1. Number of points should be scalable to allow testing various sized computers in a reasonable amount of time.
2. The results of the tests should be easily, visually recognizable as correct or incorrect. Furthermore, quantitative results should be easily interpretable.
3. Grids should be variable to encompass normal cases and all known pathologies (this is particularly important for CFD).
4. The fields defined over the grids should also encompass normal cases and all known pathologies.
5. The suite should be easily distributable to most hardware/software platforms.
6. The suite should not be dependent on language or word size.
7. The size (in bytes) of the distribution should be minimized. In particular, it is unlikely that data sets themselves should be distributed, but rather the code to generate them.
8. It should be possible to run the suite in reasonable amount of time.
9. The results should give unambiguous performance information.
10. The suite should challenge current systems.

I have a few ideas for parts of a visualization test and benchmark suite. Most of these ideas are somewhat specific to CFD. Some of them might be usefully incorporated into such a suite. In no particular order, they are:

Integral Curves (particle traces):

* Trace forward in time from a point, then trace backwards in time for the same total time from the end of the first trace. The distance between the first start point and the end of the second trace is an error measure. Do this for many points in vector fields with various properties. Be sure that some of the traces pass near critical points of various types (saddles, nodes, etc.). Also, force traces through areas of high gradient.

* Run traces in a constant direction field, but change the grid and compare results using different grids. Be sure to force traces across grid boundaries and through grid singularities.

* In multi-zone data sets, start traces near grid boundaries and stop just after passing to the next grid zone. This measures the time to pass between grid zones. In at least one case, be sure to include grid singularities where the transition takes place.

Integral Surfaces [HULT90]:

* Force surfaces around saddles. This is difficult because the surfaces tend to tear.
* Integrate surfaces into vortices. This is difficult due to twisting of the surface.

Isosurfaces:

* Generate isosurfaces on a field where all isosurfaces are sets of spheres. Change the grid such that isosurfaces must be generated on all marching cube [LORE87] cases.

Cutting Planes:

* Examine results in overlapping grids to see how the overlap is handled.
* Include benchmarks for time dependent data with static grids. This will reward the de-

veloper who takes advantage of the fact that the location of the cutting plane, and thus the vertices needed and the interpolation factors, remain constant.

Interpolation:

* Design benchmarks to test interpolation time.
* Include requirements for non-linear interpolation.

Scalar Field Local Extrema:

* If the system can find local extrema at all, check for correctness.

Vector field topology:

* Place critical points in grid cells with grid singularities.
* Place critical points on computational boundaries.

Data flow environments:

* Have modules generate grids that are input to analytic function modules that generate vector and scalar fields. Place size controls on the grid generation modules. Vary grid sizes and measure system response time.

In this brief preliminary paper, I have proposed that a visualization verification/benchmark suite be developed to measure the correctness and performance of visualization algorithms and environments. Such a suite could be of great utility in bringing increased rigor to the comparison of systems and in the claims developers and vendors make. The suite, if successful, would mold the visualization system state of the art in its image, much as super-computer benchmarks have molded hardware and software development in that arena.

Acknowledgments

This work was performed at the Applied Research Branch, NASA Ames Research Center under NASA contract NAS-2-12961.

References

[BANC90] G. Bancroft, F. Merritt, T. Plessel, P. Kelaita, R. McCabe, A. Globus, "FAST: A Multi-Processing Environment for Visualization of CFD," *Proc.Visualization '90*, IEEE Computer Society, San Francisco (1990).

[BUTL89] D. M. Butler, M. H. Pendley, "A Visualization Model Based on the Mathematics of Fiber Bundles," *Computers in Physics*, September/October 1989.

[DYER90] D. S. Dyer, "A Dataflow Toolkit for Visualization," *IEEE Computer Graphics and Applications*, July 1990, pp. 60-69.

[HULT90] J. P. M. Hultquist, "Interactive Numerical Flow Visualization Using Stream Surfaces," *Computing Systems in Engineering* **1** (2-4) pp. 349-353.

[HULT91] J. P. M. Hultquist, personal communication.

[LORE87] W. E. Lorenson, H. E. Cline, "Marching Cubes: a High Resolution 3D Surface Construction Algorithm," *Computer Graphics*, Vol 21, No 4, July 1987, pp. 163-169.

[UPSON89] C. Upson, T. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, A. van Dam, "The Application Visualization System: A Computational Environment for Scientific Visualization," *IEEE Computer Graphics and Applications*, July 1989, pp. 30-41.